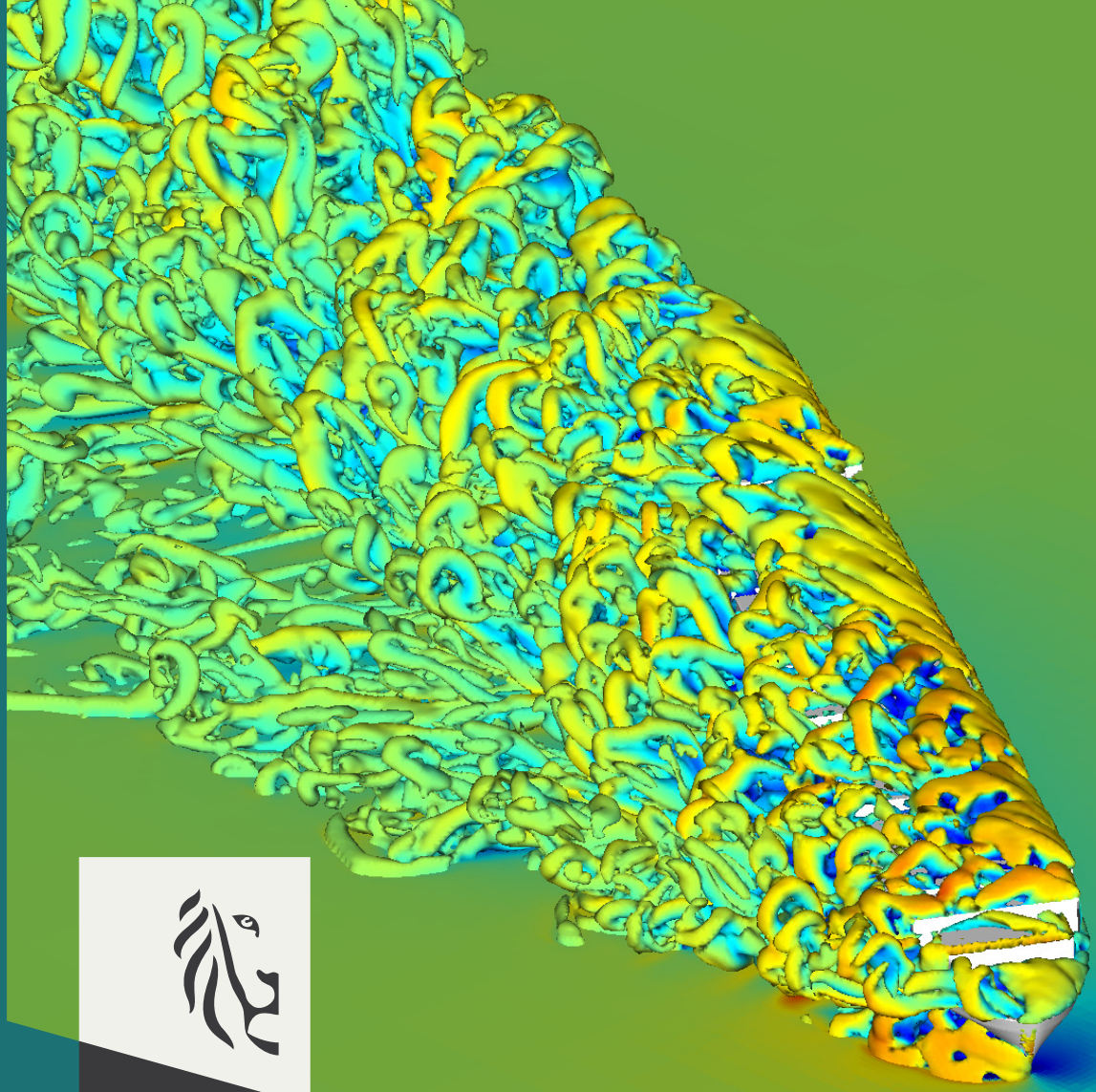


Flanders
State of
the Art

16_058_1
FHR reports

Evaluation of Gerris flow solver for the computation of wind coefficients

Optimization of geometry preparation

DEPARTMENT
MOBILITY &
PUBLIC
WORKS

www.flandershydraulicsresearch.be

Evaluation of Gerris flow solver for the computation of wind coefficients

Optimization of geometry preparation

Van Hoydonck, W.; Verwilligen, J.; López Castaño, S.; Mostaert, F.

Flanders Hydraulics Research
Flanders State of the Art

Cover figure © The Government of Flanders, Department of Mobility and Public Works, Flanders Hydraulics Research

Legal notice

Copyright and citation

D/2021/3241/017

This publication should be cited as follows:

Van Hoydonck, W.; Verwilligen, J.; López Castaño, S.; Mostaert, F. (2021). Evaluation of Gerris flow solver for the computation of wind coefficients: Optimization of geometry preparation. Version 2.0. FHR Reports, 16_058_1. Flanders Hydraulics Research: Antwerp

Document identification

Customer:	Flanders Hydraulics Research	Ref.:	WL2021R16_058_1
Keywords (3-5):	CFD, ship wind coefficients, STL geometry preprocessing, projected area computation, surface wrapping		
Knowledge domains:	Harbours and waterways > Manoeuvring behaviour > Wind > Numerical calculations		
Text (p.):	30	Appendices (p.):	0
Confidential:	No	☒ Available online	

Author(s):	Van Hoydonck, W.

Control

	Name	Signature
Revisor(s):	Verwilligen, J.; López Castaño, S.	Getekend door:Jeroen Verwilligen (Signat Getekend op:2021-01-22 14:54:07 +01:0 Reden:Ik keur dit document goed Jeroen Verwilligen Getekend door:Santiago Lopez Castaño (Getekend op:2021-02-01 14:03:15 +01:0 Reden:Ik keur dit document goed Santiago Lopez Castaño
Project leader:	Van Hoydonck, W.	Getekend door:Wim Van Hoydonck (Sign Getekend op:2021-01-22 14:04:48 +01:0 Reden:Ik keur dit document goed Wim Van Hoydonck

Approval

Head of division:	Mostaert, F.	Getekend door:Frank Mostaert (Signature Getekend op:2021-01-22 13:56:09 +01:0 Reden:Ik keur dit document goed Frank Mostaert

Abstract

The objective of project 16_058 is to evaluate the open-source Gerris flow solver for the computation of ship wind coefficients.

In the original project plan, it was foreseen to only execute a parameter variation and grid convergence study with Gerris. While this research was started in 2016, due to shifting priorities, it was only finalised in 2021. Not only was time spend for the validation, but time was also used to automate tasks that are executed before this type of Computational Fluid Dynamics (CFD) computations can be executed. These concern the determination of the reference areas of the vessel (to convert the resultant forces and moments to dimensionless coefficients) and the creation of manifold geometry from the simulator ship models. If done by hand, these last two tasks can require up to two days per ship model. The procedures documented in this report reduce this time to less than two hours.

In the present report, the automation of the tasks to prepare geometry for CFD computations are reported. A second report will present details on the validation and use of Gerris for the determination of wind coefficients, discuss a typographical error found in literature related to the experimental determination of wind coefficients and how this error affects the accuracy and reliability of CFD methods that use the results of this reference to validate their work. The second report will also present a critical review of past research related to the simulation of atmospheric boundary layers in FINE/Marine.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

CAD	Computer Aided Design
CFD	Computational Fluid Dynamics
FHR	Flanders Hydraulics Research
NURBS	Non-Uniform Rational B-Spline
RANS	Reynolds-Averaged Navier Stokes
STL	Stereolithography

Latin symbols

A	Surface area	m^2
A_F	Frontal projected area	m^2
A_I	Image area	m^2
A_L	Lateral projected area	m^2
A_P	Ship projected area	m^2
C_K	Non-dimensional moment around the x-axis	—
C_M	Non-dimensional moment around the z-axis	—
C_N	Non-dimensional moment around the y-axis	—
C_X	Non-dimensional force in the x-direction	—
$C_{X_{AF}}$	Non-dimensional force in the x-direction, using the frontal projected area A_F	—
C_Y	Non-dimensional force in the y-direction	—
C_Z	Non-dimensional force in the z-direction	—
G	Numpy array with grayscale values	—
h	height, measured positive up	m
H_S	Geometric centre of gravity of the lateral projected area, measured from the waterline	m
$\bar{H}$	Mean height of the lateral plane	m
K	Moment around the x-axis	Nm
K'	Non-dimensional moment around the x-axis	—
$\vec{L}$	Position of camera for rendering an orthographic projection	m
L_{oa}	Length over all	m
L_{pp}	Length between perpendiculars	m
M	Moment around the z-axis	Nm
N	Moment around the y-axis	Nm
n	number of vertices	—
N'	Non-dimensional moment around the y-axis	—
q	Dynamic pressure: $\frac{1}{2}\rho U^2$	Pa
$\vec{R}$	Orientation of camera	°
R_x	Horizontal resolution of image	pixels
$R_{x,0}$	Initial horizontal resolution of image in grid convergence study	pixels
$R_{x,F}$	Horizontal resolution of frontal image	pixels
$R_{x,L}$	Horizontal resolution of lateral image	pixels

R_y	Vertical resolution of frontal and lateral images	pixels
S_g	Sum of grayscale values in an array	$-$
S_x	Horizontal size of image	m
$S_{x,F}$	Horizontal size of frontal image	m
S_y	Vertical size of images	m
T	Draft	m
U_a	Apparent wind velocity	$\mathrm{m/s}$
U_s	Ship velocity	$\mathrm{m/s}$
U_w	Wind velocity	$\mathrm{m/s}$
X	Force in the x-direction	N
X'	Non-dimensional force in the x-direction	$-$
X_0	X-axis of earth-fixed axes system pointing North	$-$
Y	Force in the y-direction	N
Y'	Non-dimensional force in the y-direction	$-$
Y_0	Y-axis of earth-fixed axes system pointing East	$-$
Z	Force in the z-direction	N
Δx	Size of a pixel	m

Greek symbols

β	Drift angle: arctan $\frac{-v}{u}$	°
γ	Angle between U_a and U_s	°
ρ	Fluid (air) density	$\mathrm{kg/m^3}$
φ	Angle of relative wind, 0° is head wind	°

Subscripts

F	Frontal
L	Lateral

1 Introduction

1.1 Computation of wind coefficients using FINE/Marine

In the recent past, the CFD software suite FINE/Marine has been used to compute improved wind coefficients of the Tripoli estuary vessel after feedback from the skippers during real-time simulation trials (Van Hoydonck *et al.*, 2015b). The improved wind coefficients gave acceptable results, but a very long computing time was required (approximately one month cluster time). As a consequence, this type of computations cannot be executed on a regular basis to improve existing wind coefficients or create new wind coefficients for new simulator vessels. In another project (Van Hoydonck *et al.*, 2016) the wind field on the leeward side of a roro ship was computed using FINE/Marine to evaluate the shelter effect of a moored vessel on the manoeuvrability of another vessel. Multiple days (both cluster time and man hours) were required to generate a suitable wind field. The origin of the long turn-around time (discussed in more detail in the paragraphs that follow) of these computations is two-fold:

- computing times were long, and
- a significant number of man-hours was needed for the preparation of the computations.

The long computing times have two principle causes: a) the inclusion of viscosity in the governing (Navier-Stokes) equations next to the pressure and momentum equations, and b) the unsteadiness of the problem. The total force experienced by an object exposed to a relative wind field is a combination of pressure forces (acting perpendicular to the surfaces of the object) and viscous forces (acting parallel to the surfaces of the object). To reliably compute the viscous forces, a very large number of very small cells is required close to the surface of the object[1] which increases the computation time significantly.

For most vessels, the superstructure consists of a set of blunt objects stacked next to and on top of each other that results in massive separation regions on the leeward side of said objects. Sharp edges make the problem unsteady: vortices are shed periodically from these parts. To get the best possible result, one should use a time-accurate (at least second order in time) solver instead of a solver that is only first-order accurate in time. The time-accuracy requirement poses stricter demands on the maximum time step that can be used, with a consequence that computing times will increase further. In industrial applications, Reynolds-Averaged Navier Stokes (RANS) solvers are most commonly used for this type of application. Apart from the momentum and pressure equations, turbulence models are required for closure of the equations. The need for a turbulence model in RANS solvers increases the problem even further, because the specific turbulence model used will have an effect on the results as well.

Apart from the long computing times for this type of CFD computation, the amount of man-hours required for the setup of the computations should not be underestimated either. For a single set of coefficients (from $\varphi = 0°$ to 180°) with a 10° increment, 19 computations are required. If the CFD tool supports sliding meshes, a computational domain consisting of two separate domains (one large rectangular hexahedron and a smaller cylindrical domain in which the ship is located) can be used where the inner cylindrical domain can be rotated around the z-axis to change the wind incidence angle without creating a new mesh. This feature was however not yet available in FINE/Marine when the coefficients of the Tripoli were initially computed, hence 19 different grids were created.

[1] This can be 50% of the total number of cells used to discretize the complete domain.

During the course of the computations, (some of) the resulting grid (or grids) may have to be altered capture the wake regions close to the ship with sufficient detail. The use of automatic adaptive grid refinement can reduce the turn around time in this case by refining the grid during the computation based on e.g. the magnitude of vorticity in the wake.

Not only the step of creating the computational grid(s) may take up a significant amount of time, but also the first step, where the initial (dirty) geometry is transformed into a watertight manifold shape that has a clear distinction between inside and outside. For the determination of the wind coefficients of the Tripoli (Van Hoydonck *et al.*, 2015b), creating a watertight geometry required multiple days of editing in Blender. At that time, Hexpress was also not capable of meshing the resulting watertight hull geometry: some simplifications had to be made near the deck at the bow and stern: the area below the railing had to be filled. This was also necessary for the initial evaluation (Van Hoydonck *et al.*, 2015a), where the volume above the deck at the bow was filled up to the height of the railing. With the advance of time, new tools have been developed that can wrap a 3D geometry with a manifold mesh that makes the resulting geometry suitable for CFD computations, as discussed in Chapter 3.

1.2 Wind coefficient data used in the simulator

For most of the existing mathematical models of vessels currently available in the simulator at FHR, the origin of the wind coefficient data used is unknown. Some were copied from vessels with a similar shape (but different absolute dimensions), others are copied from similar vessels found in literature. For most of them, the exact origin is unknown with as a result that it is unknown if the data used is suitable in the first place[2].

Some of the mathematical models contain multiple configurations where a single parameter such as the draft is changed. Also, for some container and bulk cargo ships, there are multiple configurations present where only the shape of the cargo is changed (for example an even distribution of containers over the full length of the vessel, or a more step-wise distribution, with fewer containers near the bow). Without a large database of wind coefficients for all possible classes of ships (either computed using CFD or as a result of extensive wind tunnel measurements), it is difficult to know how wind coefficients should be changed to reflect a change in e.g. the container configuration for a specific case.

It should also be noted that the effect of wind is not always very important (or, not the most important part of a simulation). The forces experienced by vessels due to the proximity of banks, the bottom of the waterway or other ships may be significantly larger in magnitude than wind effects. On the other hand, there are situations where wind may be a limiting factor in the manoeuvrability of vessels (for example, at low speed during manoeuvres in a harbour in high wind conditions). At the moment only the drag force, lateral force and yawing moment are used in the simulator: this is sufficient for a mathematical model with three degrees of freedom in the horizontal plane. For the upgrade of the simulator mathematical models to six degrees of freedom, the inclusion of the roll moment due to (lateral) wind is a logical extension. The vertical (lifting) force and the pitch moment are generally less important but could be included if the coefficients are obtained from CFD computations because their computation has no additional overhead.

In the simulators at FHR, each variant of a ship has a separate `load_*.xml` file that contains data about the general characteristics of the ship, the wind coefficients table and related longitudinal and

[2] Although it is perfectly possible to add references in comments in `xml` files, the `load_*xml` files generally do not contain such information.

lateral windage areas, a wave table, and the type of mathematical models for bank suction and squat behaviour. For example, for the Tripoli estuary vessel, six load files are available with drafts between 2.52 m and 4.5 m. These all have the same wind coefficients, but they do have unique windage area values that correspond to the different drafts. If the draft of a vessel changes (and nothing else), the average height of the vessel that is exposed to wind changes as well. In a uniform wind field, the pressure on the superstructure will be (close to) proportional to the exposed area. When the wind field is non-uniform, this is not the case: the distribution of pressure (and hence the coefficients derived from it) will be a function of both the reference area and a characteristic height relative to the reference height of the velocity profile. As an example, a long vessel with a limited height with the same lateral area as another shorter but higher vessel, will have a different absolute value for the pressure acting on the lateral area.

1.3 Reference systems and definition of coefficients

When a vessel moves in a still atmosphere, the superstructure is exposed to a uniform relative wind equal to its own velocity. When the vessel is not moving in the presence of wind, the vertical wind profile will be highly non-uniform due to the viscous interaction between the moving air and the earth. For the combination of these two, when the vessel sails in the presence of wind, the total wind load consists of the wind load due to the motion of the ship in addition to the wind load due to the non-uniform atmospheric velocity (Andersen, 2013). Hence, two sets of wind coefficients are required to compute the total load: one set determined for a uniform wind profile, and a second set determined using a suitable atmospheric profile. In most nautical simulation applications, this distinction is not made: the total relative velocity U_a is computed by vector subtraction of the ships' motion U_s from the atmospheric wind U_w (see Fig. 1),

$$\vec{U}_a = \vec{U}_w - \vec{U}_s. \tag{1}$$

The relative wind angle φ can be computed[3] from the drift angle β and the three velocity components U_a, U_w and U_s. The total wind loads are obtained by multiplying the coefficient values interpolated at the φ with the reference dynamic pressure q and suitable reference areas and lengths.

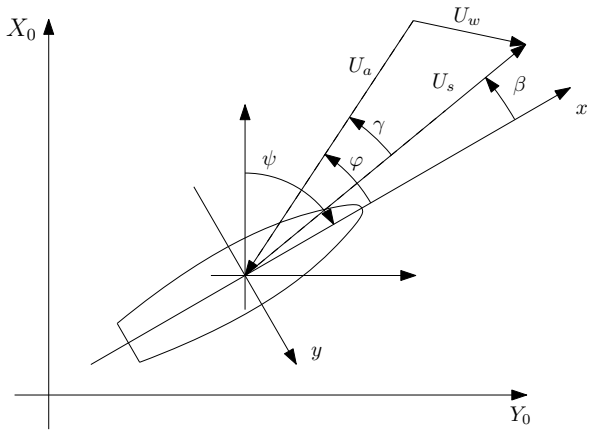


Figure 1 – Definition of the apparent wind vector U_a and the relative wind angle φ.

[3] For the current investigation, φ is set directly.

The dimensionless wind load coefficients are defined as (Blendermann, 2013)

$$C_X = \frac{X}{qA_L}, \tag{2}$$

$$C_Y = \frac{Y}{qA_L}, \tag{3}$$

$$C_Z = \frac{Z}{qA_L}, \tag{4}$$

$$C_K = \frac{K}{qA_L\bar{H}}, \tag{5}$$

$$C_M = \frac{M}{qA_L L_{oa}}, \tag{6}$$

$$C_N = \frac{N}{qA_L L_{oa}}, \tag{7}$$

where $q = \frac{1}{2}\rho {U_a}^2$ is the dynamic pressure, A_L the lateral projected area, L_{oa} the reference length of the lateral plane and $\bar{H}$ is the mean height of the lateral plane,

$$\bar{H} = \frac{A_L}{L_{oa}}. \tag{8}$$

Blendermann (2013) relates longitudinal force to the frontal projected area,

$$C_{X_{AF}} = C_X \frac{A_L}{A_F}. \tag{9}$$

Fujiwara and Nimura (2005) and Ueno *et al.* (2012) use Eqs. 9, 3, 5 and 7 to derive their mathematical model of the wind loads.

However, the marine literature is not entirely uniform when it comes to defining wind coefficients. For example, Andersen (2007) and Wagner (2005) use the following definitions for C_X, C_Y, C_N and C_K:

$$C_X = \frac{X}{qA_F}, \tag{10}$$

$$C_Y = \frac{Y}{qA_L}, \tag{11}$$

$$C_K = \frac{K}{qA_L H_S}, \tag{12}$$

$$C_N = \frac{N}{qA_L L_{oa}}, \tag{13}$$

where H_S is the geometric centre of gravity of the lateral projected area measured from the waterline (Wagner, 2005). Both H_S and $\bar{H}$ are reference heights, and for a rectangle with base L_{oa} and height h, they are related as follows:

$$H_S = \frac{\bar{H}}{2}. \tag{14}$$

Dimensionless values computed with Eqs. 10 to 13 are preferred when experimental values are to be used for a ship that is not geometrically similar to the one for which the coefficients were determined (Andersen, 2013). To allow for a comparison of difference configurations of the same ship,

coefficients that make use of reference dimensions that are independent of the reference areas A_L and A_F or quantities derived from these reference areas are preferred (Andersen, 2007; Andersen, 2013; Wagner, 2005):

$$X' = \frac{X}{qL_{pp}{}^2}, \tag{15}$$

$$Y' = \frac{Y}{qL_{pp}{}^2}, \tag{16}$$

$$K' = \frac{K}{qL_{pp}{}^3}, \tag{17}$$

$$N' = \frac{N}{qL_{pp}{}^3}. \tag{18}$$

1.4 Problem summary

From the computational point of view, there is a question of efficiency that needs to be addressed so that a timely answer can be given to questions of the nautical researchers related to the adaptation of existing and creation of new wind coefficients for simulator vessels. This holds true for both the required time for preparing Computer Aided Design (CAD) models and the time to configure and execute computations.

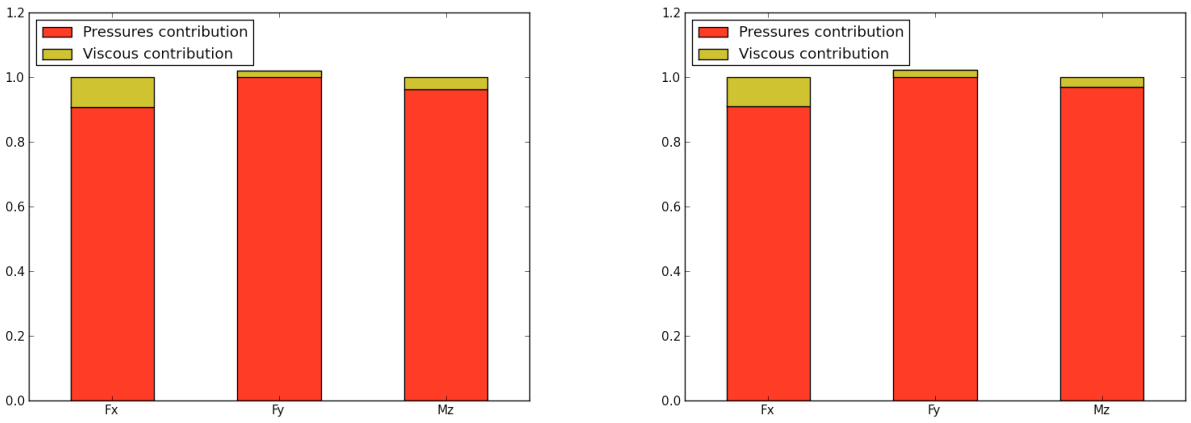
From the side of the nautical researchers at FHR, there is a desire to reduce some of the guess work that is currently involved in creating wind coefficients for new simulator vessels or adapting them based on feedback from pilots. Without knowing the magnitude of the influence of configuration changes on the wind coefficient values, it is difficult to determine if such changes even require modifications to existing wind coefficient data in the first place.

1.5 Proposed solution

1.5.1 Contributions of viscosity and pressure

An analysis of the components of the resultant forces and moments from $\varphi = 0°$ to 180° has shown that the contribution of the pressure to the resultant forces and moments is by far the largest (Van Hoydonck *et al.*, 2015b). The relative contribution of viscous stresses is rather small, as can be seen in the two graphs in Fig. 2 for the case of head wind ($\varphi = 0°$). For the drag, the viscous component is approximately 10% while for the other two components, it is smaller than 3%.

This result (and the complete analysis presented in Van Hoydonck *et al.* (2015b)) suggests that it may be sufficient to use a solver that neglects the contribution of the viscosity: using an Euler solver instead of a Navier-Stokes solver. As stated before, this can have a significant impact on the computing time as this reduces the number of cells (required for the resolution of the boundary layer) significantly.



(a) Steady computation.

(b) Unsteady computation.

Figure 2 – Relative contribution of the viscous and pressure components to the total forces and moments for a scale model of a container ship (without containers on deck) in a head wind (Van Hoydonck *et al.*, 2015b).

1.5.2 Grid generation

The second issue that contributes to the high turn-around time is the pre-processing time required to generate numerical grids. The solver should ideally be able to adapt its grid to the problem at hand in a completely automatic way, using simple logical rules.

1.5.3 CFD Solver

To the knowledge of the author, there is only one CFD solver that addresses these points: the open-source Gerris flow solver (discussed in more detail in section 2.5).

An initial evaluation of Gerris has shown that for the Tripoli, the shape of the resultant lateral force is very similar (although somewhat higher) to the one computed using FINE/Marine (see Fig. 3), in a time frame that was almost two orders of magnitude shorter than using FINE/Marine. The latter results were computed using an atmospheric boundary layer at the inlet whereas the Gerris results were computed using a uniform wind field at the inlet, which explains the higher C_Y values for Gerris as compared to FINE/Marine.

There is a significant difference in the longitudinal force (especially above 45 degrees) for which an explanation should be found. However, this does not mean that Gerris should be discarded if discrepancies remain, because as stated above, the velocity profile at the position of the vessel in the computational domain for the FINE/Marine result is actually unknown. Both for the Gerris results and the FINE/Marine results, the period of the C_X graph is proportional to $\sin(4\varphi)$. In literature, this type of graph for C_X is associated with wind tunnel tests of passenger ships (Blendermann, 2013; Fujiwara and Nimura, 2005; Ueno *et al.*, 2012). This may mean that simplifying the containers on the deck of the CAD geometry of the Tripoli to a single block is one simplification too much; in other words, the gaps between the container bays may need to be preserved in order to get a correct graph for C_X. This is also concluded by Janssen *et al.* (2017), where wind coefficients computed using four different geometric approximations of a container ship are compared with experimental results. They found a significant improvement in the results when the gaps between container stacks are included in the ship geometry.

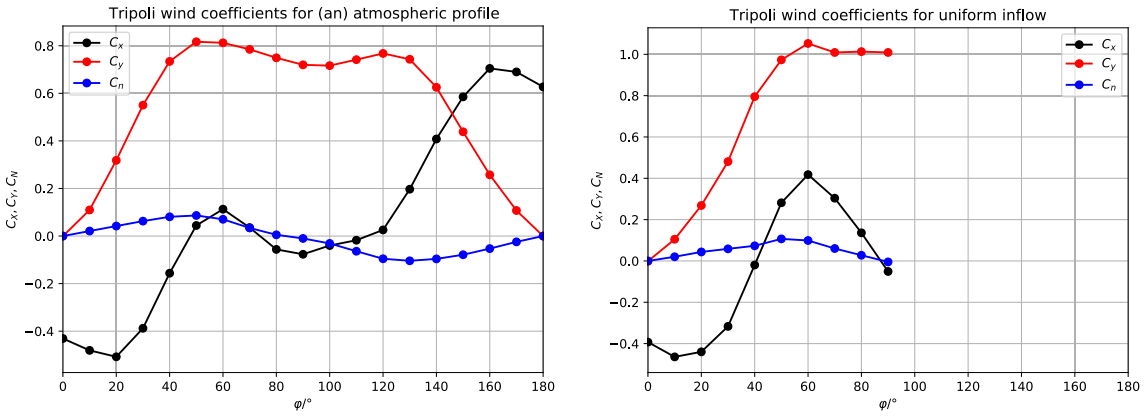


Figure 3 – Wind coefficients of the Tripoli estuary vessel computed using (a) FINE/Marine with an atmospheric boundary layer and (b) Gerris with a uniform wind field.

Although it is possible to set a non-uniform velocity profile at the inlet in Gerris, this will not be pursued here for multiple reasons:

- there is no guarantee that an arbitrary velocity profile defined at the inlet remains constant throughout the computational domain (Blocken *et al.*, 2007; Richards and Norris, 2012);
- wind tunnel tests are often performed in a uniform wind field (Blendermann, 1993);
- the addition of a non-uniform velocity profile at the inlet in Gerris means that the cell count will be enlarged to resolve the profile until it arrives at the vessel. Due to the use of square cells, Gerris has a disadvantage over a solver that uses hexahedral cells of arbitrary aspect ratio. The impact on the required computation time is significant.
- it is possible to convert coefficients derived or obtained for a specific vertical velocity profile to a set of coefficients that correspond to a different wind field (Blendermann, 2013): as long as the velocity field is known for which the wind coefficients are valid, the coefficients can be converted to a different velocity field. Using a uniform velocity field to determine the wind coefficients and automatically converting them at the start of a simulation to correspond to the selected atmospheric profile seems like a good compromise.

1.6 Report contents

In this report, research is documented related to the preparation phase of a CFD computation to determine wind coefficients on a ship. Automatic computation of the reference areas from the input STL file will be discussed and a tools are documented to create a manifold geometry from the input STL file. The topics related to the fluid dynamics side of the research will be documented in a subsequent report.

2 Projected Surface Area Computation

2.1 Introduction

This chapter describes a method to automatically compute the frontal and lateral windage areas of vessels used in the simulator at FHR. These reference areas are required to convert the output of Gerris to the coefficient forms as presented in literature (Blendermann, 2013). Due to the large number of vessels and possibly large number of configurations per vessel (e.g. with a different container stacking, different draft, …), the total number of area values that must be computed is very large. This calls for a method where the computation itself is automated, without requiring manual intervention from the user.

Initially, focus was on developing an algorithm where a contour polygon was constructed around the part of the hull above water. Once this polygon is constructed, the area A can be computed exactly from the positions of the vertices of the polygon:

$$A = \sum_{k=0}^{n} \frac{(x_{k+1} + x_k)(y_{k+1} - y_k)}{2}, \tag{19}$$

where n is the number of vertices, (x_k, y_k) is the k-th vertex and $(x_{k+1}, y_{k+1}) = (x_0, y_0)$: the first and last vertex are co-located.

Construction of the polygon was attempted by scaling the model in the normal direction and comparing face normals of adjacent faces. If these have the opposite sign, the edge that joins the faces is located at the boundary of the shape. This works fairly well for non-manifold geometries, but once holes are present or the mesh is manifold (i.e, an edge is shared by either one or more than two faces), detection of the boundary edges becomes difficult. Also, the 3D models generally consist of multiple disconnected parts that overlap which means that intersections between the parts must be computed as well. The algorithm described above works for simple cases, but to cover all possible corner cases, insufficient time was available for a robust implementation. This means that with the current state of the algorithm, manual intervention is required.

An alternative approximate method was then devised that will be explained in the next section.

2.2 Automatic approximate projected area computation

2.2.1 Introduction

By creating an Cartesian (structured) grid that overlays the ship geometry in for example the lateral or front view, the area can be estimated by determining the number of cells in the grid that are partially or fully covered by the hull geometry. An initial implementation was created in Matlab. Due to the high resolution required to capture small details, the required computing time is very high (up to 750 seconds for a grid with 1000 cells in the x-direction for one particular case).

This algorithm can only provide an estimate of the projected area and hence, one needs to ensure that the obtained value is sufficiently close to the actual projected area. If a certain ship geometry

contains very small details, the computed area will fluctuate as long as the grid is coarser than the smallest geometric details.

An alternative implementation of this algorithm was created in Blender (using Python) where instead of constructing an overlay grid and checking the coverage of individual grid cells, an orthographic black and white render of the hull geometry is created and the number of black pixels is counted. If the hull geometry fits the rendered image exactly, the area of an individual pixel is known and the total ship area equals the sum of the black pixels. By activating an option in Blender to use oversampling (or anti-aliasing), pixels can be coloured grey as well based on the relative coverage of the ship geometry at the pixel location. This way, it is possible to include the influence of small details on the area computation without using a grid that is fine enough to actually resolve the details completely.

For the implementation, the STL geometries of the vessel `DFDS_JinLing_235_330` at a draft of 7 m (Fig. 4), a configuration of the Tripoli estuary vessel and the Barzan container ship are used.

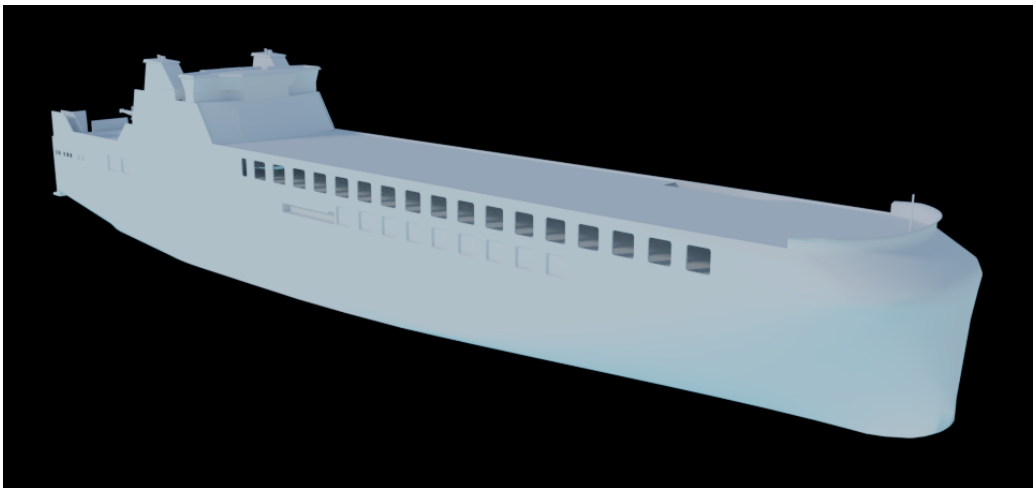


Figure 4 – CAD geometry of the DFDS JinLing car carrier as used in the simulator of FHR.

2.2.2 Algorithm implementation

Starting from a set of user options, a command is generated with which a Python script is executed inside Blender. The user options are:

- STL file to compute reference areas for;
- Draft at which to compute the reference area;
- horizontal resolution for the lateral view;
- Anti-aliasing setting;
- use of nodes for rendering;
- base name to use for output.

For the DFDS JinLing, a typical command is displayed in Listing 1 which defines:

- the Blender executable to use (`"C:/Program Files/Blender Foundation/Blender/blender.exe"`);
- and how it is run (`--background`);
- that it should execute the supplied Python script (`"C:/Algemene gegevens/16_058/ship_area_computation/bpy_generate_views.py"`).

Listing 1 – DOS command to compute the surface area of the DFDS JinLing at a horizontal resolution of 3200 pixels.

```
"C:/Program Files/Blender Foundation/Blender/blender.exe" --background^
--python "C:/Algemene gegevens/16_058/ship_area_computation/bpy_generate_views.py"^
-- 7 3200 "C:/Algemene gegevens/16_058/ship_area_computation/DFDS_JinLing_235_330.stl"^
no_nodes "C:/Algemene gegevens/16_058/ship_area_computation/DFDS_JinLing_235_330_03200px_AA8_T7"
```

The whitespace separated list of options after the double dash are passed to this Python script and these set:

- the draft (7 m);
- the horizontal size ($R_{x,L}$) for the lateral area computation (3200);
- the input STL file (`"C:/Algemene gegevens/16_058/ship_area_computation/DFDS_JinLing_235_330.stl"`)
- that nodes should not be used (`no_nodes`);
- that the basename for the output files is `"C:/Algemene gegevens/16_058/ship_area_computation/DFDS_JinLing_235_330_03200px_AA8_T7"`;
- the amount of anti-aliasing samples per pixel should be set to 8. Valid values are 0, 5, 8, 11 and 16.

2.2.3 Blender steps

Using the base name for the output, a log file is opened to which useful input and output values are written for later use such as the draft, the resolution, the STL file and the filenames that contain the lateral and frontal black and white orthographic views of the vessel at the requested draft.

2.2.3.1 Importing the geometry

First a shadeless black material is created for the geometry. The STL file[4] is imported and the newly created material is assigned to the ship object.

Using the bounding box of the ship and its dimensions, dictionaries are constructed with geometrical data of the ship size ($|x|, |y|, |z|$), its extends and the centre location ($\bar{x}, \bar{y}, \bar{z}$). For the vertical extents, the supplied draft value T is required: if the smallest z-coordinate is less than 0, then $z_{min} = \max(T, z_{min})$, else, the (positive) draft value is used.

Once the extrema of the ship are known, position and orientation for the cameras to render the lateral and frontal orthographic projections are determined where a distinction is made based on the orientation of the vessel. When the size in the x-direction is larger than the size in the y-direction, the following arrays are used:

$$\vec{L}_L = (\bar{x}, -|x|, \bar{z}) \,, \tag{20}$$
$$\vec{R}_L = (-90°, 180°, 180°) \,, \tag{21}$$
$$\vec{L}_F = (|x|, \bar{y}, \bar{z}) \,, \tag{22}$$
$$\vec{R}_F = (-90°, 180°, -90°) \,, \tag{23}$$

[4] Blender can handle both binary and textual STL files in a transparent way.

otherwise, the following values are used:

$$\vec{L}_L = (|y|, \bar{y}, \bar{z}), \tag{24}$$
$$\vec{R}_L = (-90°, 180°, -90°), \tag{25}$$
$$\vec{L}_F = (\bar{x}, |y|, \bar{z}), \tag{26}$$
$$\vec{R}_F = (-90°, 180°, 0°). \tag{27}$$

Finally, the maximum (d_{max}) and minimum (d_{min}) horizontal dimensions are set.

2.2.3.2 Preparing the scene for rendering

The (square) pixel size is determined from the maximum horizontal dimension and the requested horizontal image resolution:

$$\Delta x = \frac{d_{max}}{R_{x,L}}. \tag{28}$$

The vertical resolution of the images in pixels (R_y) is determined from the ship vertical size $|z|$,

$$R_y = \left\lceil \frac{|z|}{\Delta x} \right\rceil, \tag{29}$$

which may be more than the actual ship vertical size.

For the frontal area computation, the horizontal resolution $R_{x,F}$ is determined:

$$R_{x,F} = \left\lceil \frac{d_{min}}{\Delta x} \right\rceil, \tag{30}$$

which, due to rounding, may be slightly more than the actual ship width.

The vertical size of the lateral and frontal images S_y is computed using the pixel size and the vertical resolution,

$$S_y = \Delta x R_y, \tag{31}$$

while the horizontal size of the frontal images $S_{x,F}$ follows from

$$S_{x,F} = \Delta x R_{x,F}. \tag{32}$$

S_y and $S_{x,F}$ are in general slightly larger than $|z|$ and d_{min} because the vertical to horizontal size ratio is not guaranteed to be an integer value.

Due to the increase in vertical size, the vertical camera location must be increased slightly with

$$\Delta z = 0.5(S_y - |z|). \tag{33}$$

It is assumed that for the frontal image, the horizontal shift of the camera is zero because the vessels are assumed symmetric with respect to the butt line (xz-plane).

Two orthographic camera objects are created: one for the lateral view and one for the frontal view. The orthographic scale (`ortho_scale`) is the important variable. It represents the maximum dimension (in scene units) of the portion of the space captured by the camera. For the lateral view, this is

equal to the maximum size of the vessel, while for the frontal view, it is the maximum of the adjusted vertical size and the minimum horizontal ship size.

The world (background) horizon and ambient colours are set to white. The vertical resolution of both images is R_y. The horizontal resolution depends on the actual image that is generated. For the lateral view, it is the requested resolution $R_{x,L}$, while for the frontal view, it is $R_{x,F}$.

Image settings are added as well: the colour mode is set to `RGBA` with a depth of 8 bit per channel. Compositing and the sequencer are disabled while image overwriting is enabled. Antialiasing settings are set based on what the user requested. For complex geometry with small details, a value of 8 seems to give a good balance between render speed and convergence speed (see later).

2.2.3.3 Render scene

Once the scene preparation is finished, it is rendered and the output is written to a png file. The actual filename is returned and used to load the image in memory again (as a list of pixels). Direct manipulation of the render result in memory without writing the data to disk first does not seem to work reliably in Blender 2.79.

2.2.3.4 Ship area computation

The rendered image (black and white) is converted into a numpy array G where only each fourth pixel (due to the four channel `RGBA` image format) is retained, starting at the first one). This means that for every pixel, there is 1 float value (between 0 and 1) that represents its greyscale colour.

The sum of this array S_g is computed from the size (in pixels) of the image minus the sum of the greyscale values (here for the lateral image):

$$S_g = R_x R_y - \sum G. \tag{34}$$

The area covered by the image A_I follows from the pixel size and the computed horizontal and vertical dimensions,

$$A_I = S_x S_y \Delta x^2, \tag{35}$$

and finally, the ship projected area A_P follows as

$$A_P = A_I * S_g/(S_x S_y), \tag{36}$$

which can be computed directly from S_g and Δx,

$$A_P = \Delta x^2 S_g. \tag{37}$$

This routine is called twice, first for A_L and then for A_F. The computed ship area is written to the log file as well.

2.3 Grid convergence study

To ensure that the above algorithm is implemented correctly (i.e. it computes the correct area) and that it is used at a sufficient resolution, it is used to compute the frontal and lateral surface areas of simple geometries for which the surface area can be determined analytically. The influence of the antialiasing setting on the computed area will be determined as well.

2.3.1 Simple box with analytical surface areas

The lateral and frontal projected surface areas of the box in Fig. 5 are

$$A_L = \frac{20+15}{2}40 = 700\,\mathrm{m}^2, \tag{38}$$

$$A_F = \frac{15+20}{2}20 = 350\,\mathrm{m}^2. \tag{39}$$

An array with resolution values is generated with common ratio of approximately $\sqrt{2}$ (geometric progression) starting at 50 and ending at 6400:

$$R_{x,L} = [50, 70, 100, 141, 200, 282, 400, 564, 800, 1128, 1600, 2256, 3200, 4512, 6400]. \tag{40}$$

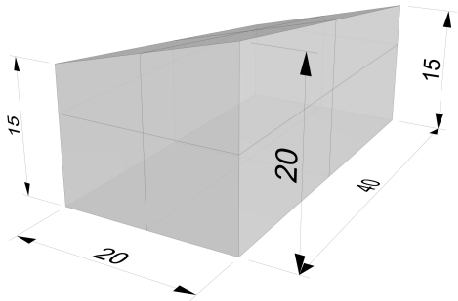
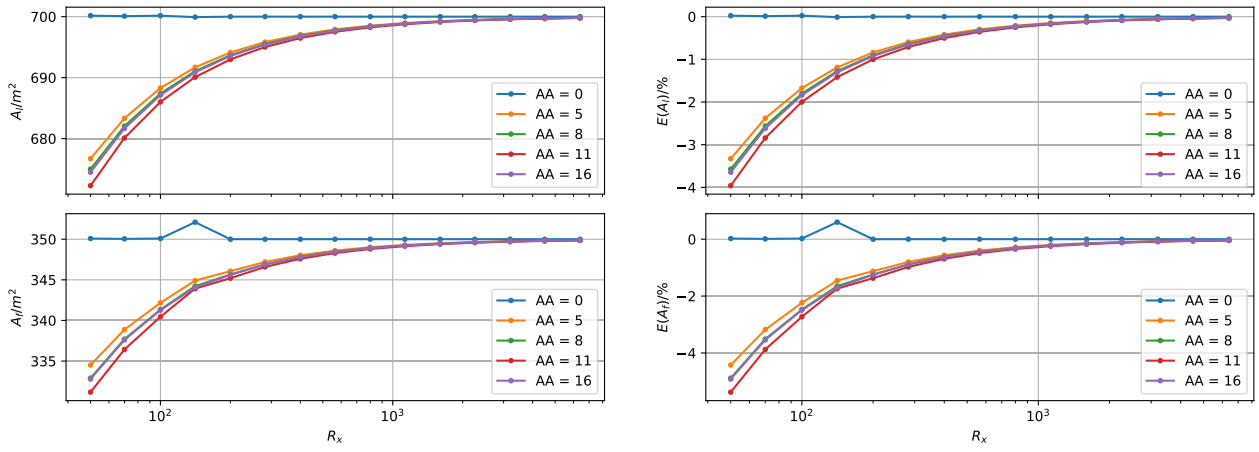


Figure 5 – Dimensions of a simple box for verification of the surface area computation.

For all values of the anti-aliasing parameter (0, 5, 8, 11 and 16), the computed lateral and longitudinal surface areas are displayed in Fig. 6a. There is a clear difference between results obtained with anti-aliasing enabled, and results obtained with this feature disabled (AA = 0). The former curves (AA enabled) converge very gradually towards the analytical surface areas, whereas the results obtained with AA disabled are exact with resolutions of 800 pixels and higher. Inspection of the relative errors (Fig. 6b), the curves with AA enabled show a relative error smaller than 0.1% for resolutions starting at 3200 pixels. For relative errors smaller than 1%, a horizontal resolution of 282 is required.



(a) Absolute values.

(b) Relative errors.

Figure 6 – Convergence of surface area computation as a function of resolution for the simple box.

A Python run that computes the frontal and lateral surface area at 15 different resolutions takes between 71 s (AA = 0) and 82 s (AA = 16) on the author's laptop. A breakdown of the timings is shown in Fig. 7 where for each resolution, the execution times of the Blender Python scripts are displayed as a function of the resolution. This figure also displays the slope of a quadratic equation that approximates the slope of the curves for high resolutions. This means that for high resolutions, the computing time scales with the square of the grid resolution. At (very) coarse resolutions, the computing time ($\approx$ 0.14 s) is basically independent of the resolution: the steps to setup the scene are independent of the image resolution and dominate the total time.

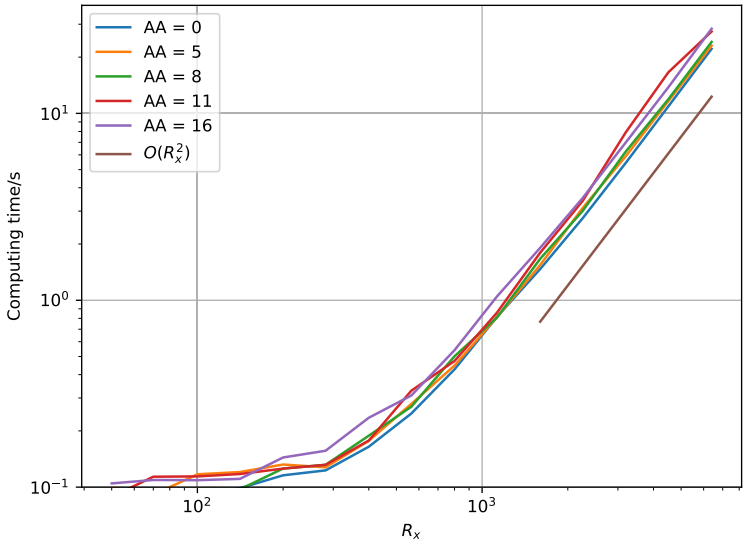


Figure 7 – Computing time as a function of resolution.

For the simple geometry used here, it is clear that the anti-aliasing settings do not improve the speed of convergence. There is also little difference between the different non-zero anti-aliasing settings: higher values increase the computing time slightly.

The above exercise will be executed again with a simple ship geometry for which the frontal and lateral surface areas are known exactly. Afterwards, a ship geometry for which the surface area is not know exactly will be tested. For these cases, only the AA = 0 and AA = 8 settings will be tested.

2.3.2 Simplified Barzan hull

A simplified hull of the Barzan container ship has been used in CFD computations related to wind coefficient predictions (Van Zwijnsvoorde *et al.*, 2019). For this simple geometry (see Fig. 8), the reference areas are known exactly: $A_L = 17\,100.8\,\text{m}^2$ and $A_F = 3287.46\,\text{m}^2$.

For this case, absolute values of the lateral and frontal surface areas are shown in Fig. 9a while the relative errors of these quantities are shown in Fig. 9b. Compared to the convergence graphs of the previous geometry, the convergence now is not as smooth and quick. For the lateral area computation, for $R_{x,L} = 800$, the error is smaller than 1%. For the frontal area computation, this level of error is reached with $R_{x,L} = 200$. For the lateral area computation, convergence is monotonic, whereas for the frontal area computation, it is oscillatory.

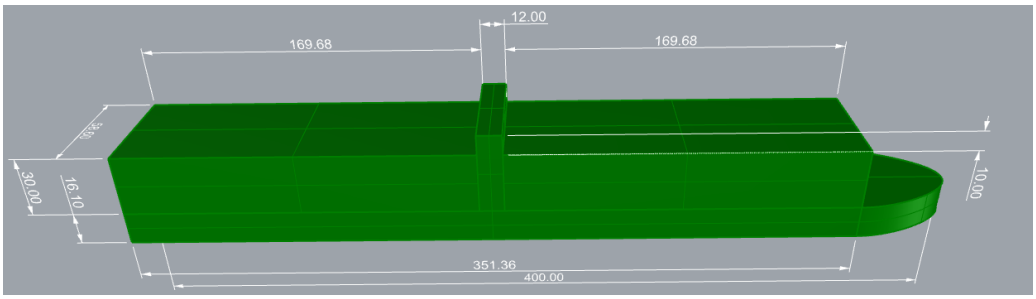
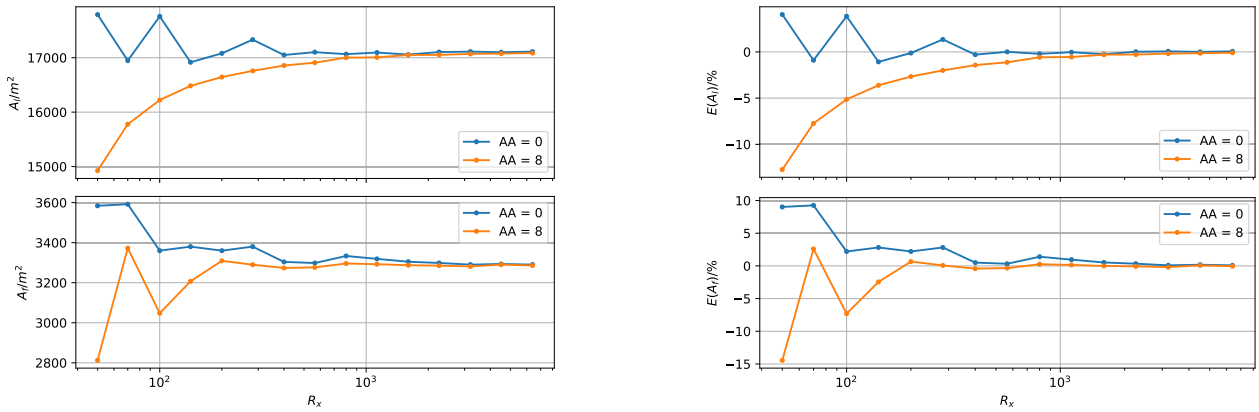


Figure 8 – Dimensions of a simplified hull form of the Barzan container vessel.



(a) Absolute values. (b) Relative errors.

Figure 9 – Convergence of surface area as a function of resolution for the Barzan hull geometry.

2.3.3 Estuary vessel Tripoli

The estuary vessel the Tripoli is used as a test case of a representative geometry as used in the simulator. A modified version of this geometry has been used in a previous CFD study to determine a set of wind coefficients (Van Hoydonck *et al.*, 2015a,b). For the current surface area computation, the actual geometry as utilised in the simulator is used –unlike the CFD computations, where the geometry was simplified significantly– which includes container stacks protruding through the bottom of the hull, see Fig. 10. The draft of the vessel is arbitrarily set to 2.5 m.

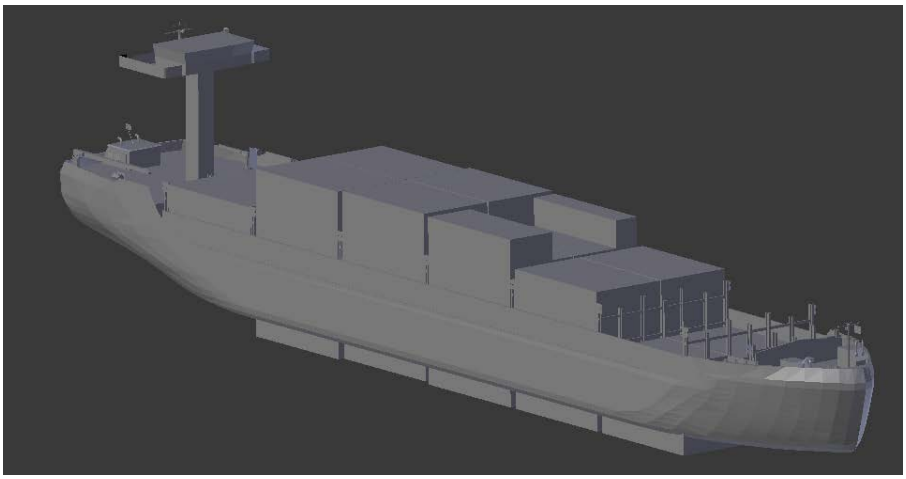


Figure 10 – Overview of the geometry of the Tripoli estuary vessel as used in the simulator.

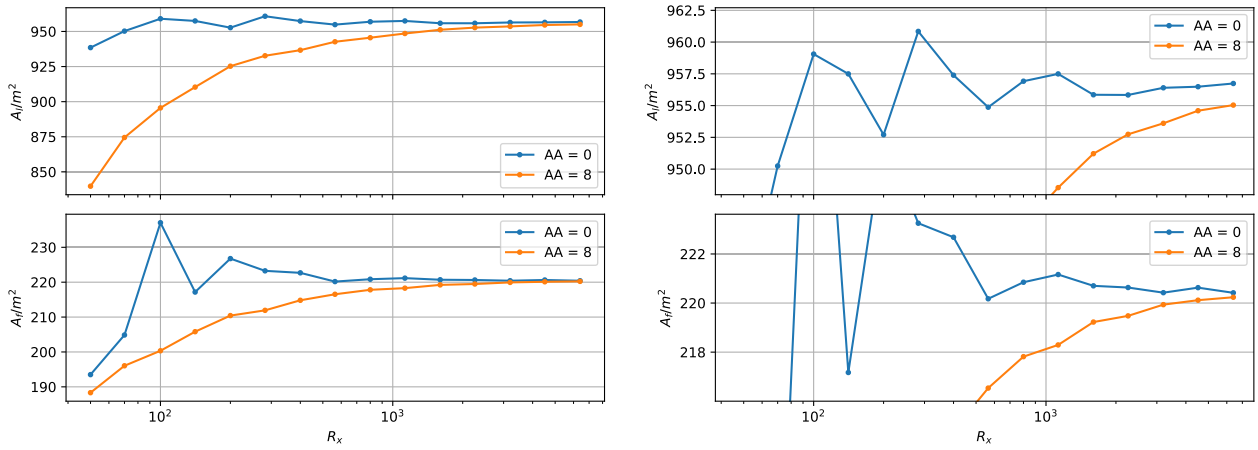
For this case, there is no reference area known, so only the absolute values of the frontal and lateral surface areas will be shown. As before, anti-aliasing is used as parameter: AA=0 and AA=8 are used. The results are shown in Fig. 12. For the lateral surface area, the average of the prediction at the finest grid equals 955.8 m^2 while for the frontal surface area, this average is 220.3 m^2.

The lateral and frontal view of the Tripoli at a resolution of 3200 pixels with anti-aliasing enabled is shown in Fig. 11.

Figure 11 – Frontal and lateral view of the Tripoli for $R_{x,L} = 3200$ and $AA = 8$, with $A_L = 953.60\,m^2$ and $A_F = 219.94\,m^2$.

Inspection of the renderings shows that at the finest level, small gaps appear for the case with anti-aliasing enabled that are not visible in the case with anti-aliasing disabled. These small details put a constraint on the minimum required grid resolution to resolve this type of features. As a consequence, it may be necessary to compute the surface areas at even finer grids.

To put this argument in perspective, the STL geometry does contain features such as flags and filled polygons that are textured with transparent textures in the simulator that all increase the total surface area of the vessel. In reality, the surface areas of these features should not be used in the surface area computation. So, by increasing the resolution of the surface area computations, one can reduce the error of the estimate, but the resulting value may contain a bias because some parts of the geometry are actually transparent and should not be included in the surface area computation.



(a) Absolute values.

(b) Zoom of Fig. 12a.

Figure 12 – Convergence of surface area as a function of resolution for the Tripoli hull geometry.

2.4 Surface area extrapolation

From the above grid convergence study, it follows that for ship geometries as used in the simulator, a fine resolution is required to capture the effect of small geometric features in the surface area computation. In this section, a sequence of successively finer grids is used to fit a function through

the computed areas that can be used to improve the surface area estimation without computing extra surface area values.

The following four-parameter symmetric sigmoidal function is used to fit the data,

$$y = d + \frac{a-d}{1+\left(\frac{x}{c}\right)^b}, \tag{41}$$

where a, b, c and d are parameters that must be determined, x is the resolution and y the computed surface area. In the limit of a very high resolution x, the value of y (the ship surface area) is equal to the parameter d. This method is applied to the data generated in the previous section for the three test geometries. For the first two geometries, the error of the extrapolated surface area will be computed and compared to the relative error of the result on the finest grid.

2.4.1 Test cube

The relative errors of the parameter d in Eq. 41 are shown in Table 1 for the lateral and frontal surface area extrapolation for all five values of the anti-aliasing settings used before as compared to the result on the finest grid (right-most column). The values in columns three to seven correspond to the errors obtained when the sigmoidal fit starts at the resolution indicated in the first row. When only high resolution results are used to estimate the surface areas (e.g. column seven), the relative error is significantly smaller (in some cases 6 orders smaller) than the relative error of the result on the finest grid (last column). By progressively including results at coarser resolutions in the extrapolation, the estimated surface area is further from the analytical value.

One can conclude from these results that it is indeed possible to fit a function through the results to try to improve the surface area estimation without computing them at extremely fine resolutions.

Table 1 – Relative errors (%) of the lateral and frontal surface areas for the test cube as a function of the anti-aliasing value for curve fits starting at progressively finer resolutions as compared to the result on the finest grid (last column).

	AA	$R_{x,0}=50$	100	200	400	800	6400
A_L	0	-7.785×10^{-4}	-7.785×10^{-4}	-6.712×10^{-4}	-4.014×10^{-5}	0.0	0.0
A_F		4.382×10^{-2}	1.352×10^{-4}	-7.511×10^{-4}	7.984×10^{-5}	0.0	0.0
A_L	5	1.180×10^{-3}	5.276×10^{-4}	4.375×10^{-4}	7.100×10^{-4}	1.063×10^{-3}	-2.578×10^{-2}
A_F		-9.654×10^{-3}	-8.640×10^{-3}	-2.101×10^{-4}	-7.595×10^{-5}	-1.055×10^{-7}	-3.536×10^{-2}
A_L	8	-8.301×10^{-4}	1.370×10^{-4}	-9.857×10^{-5}	-1.504×10^{-4}	-3.725×10^{-8}	-2.832×10^{-2}
A_F		-6.018×10^{-3}	-6.808×10^{-3}	-1.595×10^{-4}	-1.689×10^{-4}	-1.248×10^{-7}	-3.910×10^{-2}
A_L	11	-4.735×10^{-4}	3.625×10^{-4}	-2.280×10^{-5}	-9.788×10^{-5}	-3.773×10^{-8}	-3.142×10^{-2}
A_F		-1.360×10^{-2}	-1.339×10^{-3}	-4.775×10^{-6}	-1.059×10^{-4}	-1.341×10^{-7}	-4.313×10^{-2}
A_L	16	-1.040×10^{-3}	3.101×10^{-4}	-9.267×10^{-5}	-1.117×10^{-4}	-3.932×10^{-8}	-2.892×10^{-2}
A_F		-4.464×10^{-3}	-4.291×10^{-3}	1.661×10^{-5}	-6.982×10^{-5}	-1.275×10^{-7}	-3.942×10^{-2}

2.4.2 Simplified Barzan hull

The same exercise is done for the simplified hull form of the Barzan container vessel. The relative errors of the parameter d in Eq. 41 are shown in Table 2 for the lateral and frontal surface area extrapolation for the two values of the anti-aliasing settings used before (§ 2.3.2) as compared to the result on the finest grid (right-most column). Careful inspection of these error values reveals that the relative errors of the curve fit starting at $R_{x,0}=100$ for $AA=8$ gives the smallest errors. Compared

to the results obtained with the finest grid, the errors are approximately three times smaller. This is not as good as for the test cube case discussed in the previous section, which may mean that higher resolutions are required.

Table 2 – Relative errors (%) of the lateral and frontal surface areas for the simplified Barzan hull for $AA = 0$ and $AA = 8$ for curve fits starting at progressively finer resolutions as compared to the result on the finest grid.

	AA	$R_x, 0 = 50$	100	200	400	800	6400
A_L	0	1.755×10^{-1}	-4.146×10^{-2}	5.269×10^{-2}	6.763	3.411×10^{-1}	6.111×10^{-2}
A_F		5.568×10^{-1}	4.876×10^{-1}	3.042×10^{-2}	-9.824×10^{1}	9.022×10^{-2}	8.724×10^{-2}
A_L	8	-2.596×10^{-1}	2.783×10^{-2}	-6.561×10^{-2}	-8.959×10^{-2}	-9.175×10^{-2}	-9.343×10^{-2}
A_F		-4.992×10^{-1}	1.521×10^{-2}	-6.33×10^{-2}	2.640×10^{-2}	-5.232×10^{-2}	-4.259×10^{-2}

2.4.3 Tripoli surface area estimation

For the Tripoli estuary vessel, analytical values for the surface areas are not known, so only absolute values can be computed. Results are gathered in Table 3. Based on the increasing trends shown in Fig. 12b, one would expect the lateral area estimate to be higher than the value on the finest grid for $AA = 8$. Inspection of the values in Table 3 shows that this is indeed the case. The average of the ten extrapolated lateral surface area values is 956.4652 m^2, while for the frontal surface area, the average value equals 220.8162 m^2. These averaged values are closest to the extrapolated values for $R_x, 0 = 100$ with $AA = 8$.

Table 3 – Lateral and frontal surface area values estimates for the Tripoli estuary vessel for AA=0 and AA=8 for curve fits starting at progressively finer resolutions as compared to the result on the finest grid.

	AA	$R_x, 0 = 50$	100	200	400	800	6400
A_L/m^2	0	956.777	956.504	956.884	956.326	956.533	956.740
A_F/m^2		222.454	221.238	220.564	220.623	220.525	220.417
A_L/m^2	8	955.645	956.501	956.776	956.904	955.802	955.039
A_F/m^2		220.732	220.815	220.252	220.647	220.312	220.237

2.4.4 Conclusion

For the cube, it was shown that significantly more accurate results (up to 6 orders) can be obtained by using a suitable extrapolation function. However, the size of the smallest details do play a role in the extra accuracy that can be gained: for the simplified Barzan hull geometry, the improvements were at most threefold. For this geometry, the smallest errors were obtained when the function fit started at $R_{x,0} = 100$, for the case with anti-aliasing enabled ($AA = 8$). For the Tripoli, no analytical values for the surface areas are available, so only absolute values could be compared. When computing the average of the different extrapolation results, the extrapolated results for the same parameters ($R_{x,0} = 100$ and $AA = 8$) as the Barzan case are closest to the average. One could take this as a careful recommendation to compute extrapolations using these parameters. The extrapolations that were executed here show that the results at very coarse resolutions have a detrimental effect on the accuracy of the value in the limit of a very fine resolution.

2.5 Surface area computations of other ships

For a small set of ships used in the simulator at FHR, the lateral and frontal surface areas are computed with the algorithm detailed in this chapter. The coarsest resolution is set to 100, the finest to 6400 and the anti-aliasing value equals 8. The resulting estimates of the frontal and lateral surface areas are gathered in Table 4. The lateral and frontal views of these ships with a resolution of 1600 pixels are shown in Figs. 13 to 19. For each of the computed vessels, the values as found in the `load_*.xml` for some drafts are shown in the last two columns as well. For most of these ships, the differences are significant (with both LNG models as exceptions). One notable example with significant differences is the Myzaquazo, where the differences in the lateral surface area between the computed value and the reference value as used in the simulator is up to 50%. This is a container vessel, and the surface areas depend a lot on the specific configuration of the containers. To ensure that the algorithm detailed in this chapter does not contain a fundamental flaw, the lateral surface areas were estimated by tracing a rough polygon around the STL model, filling the outline with triangles and computing the sum of the areas of these triangles. Only the widest gaps between the container stacks are included, and some of the smaller equipment (such as the crane) was left out, so the values estimated as such should be smaller than values in the table. The values found as such are 831 m^2, 635 m^2 and 528 m^2 for even draft values of 1.5 m, 2.85 m and 3.65 m. Indeed, these are smaller, but close to the computed values.

Table 4 – Estimates of the lateral and frontal surface areas of several simulator models.

STL file	T/m	A_l/m^2	A_f/m^2	A_l^s/m^2	A_f^s/m^2
qmax_al_anna.stl	12.2	8663.92 ± 3.94	1861.59 ± 1.41	7866.0	1659.0
	9.6	9526.66 ± 9.02	2001.63 ± 1.89	8727.0	1788.0
qflex_mona.stl	12.0	6717.49 ± 3.53	1469.24 ± 2.19	7500.0	1550.0
	9.5	7501.22 ± 10.40	1594.38 ± 2.04	8300.0	1650.0
myzaquazo_3layers.stl	1.5	813.45 ± 1.09	110.793 ± 0.106	654.8	84.2
	2.85	632.77 ± 1.05	85.326 ± 0.081	411.8	63.5
	3.65	525.86 ± 1.33	76.199 ± 0.136	303.8	54.4
LNG_Yamal_ARC7_299_500.stl	11.78	7656.09 ± 4.08	1710.79 ± 3.96	7714.0	1751.0
	9.9	8204.99 ± 5.68	1805.42 ± 8.03	8207.0	1837.0
LNG_Conventional_300_460.stl	12.0	6755.80 ± 5.02	1408.09 ± 2.99	6719.0	1370.0
	9.5	7488.84 ± 5.56	1524.00 ± 5.79	7431.5	1480.0
DFDS_JinLing_235_330.stl	7.0	5297.58 ± 2.71	1224.90 ± 1.84	5930.4	1182.0
clementine.stl	6.5	2996.43 ± 0.87	684.62 ± 0.32	2615.0	650.0

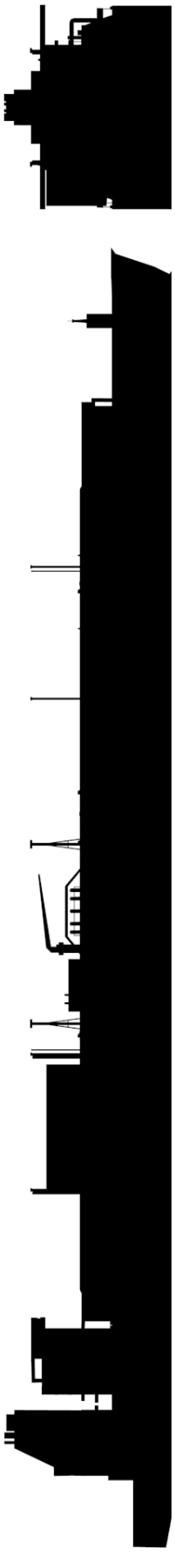


Figure 13 – Lateral and frontal view of the qmax_a1_anna.

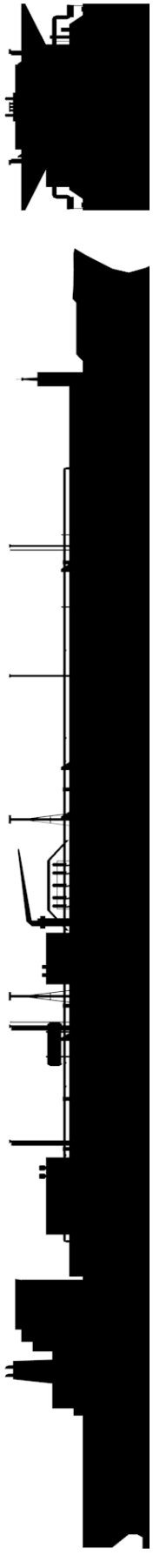


Figure 14 – Lateral and frontal view of the qflex_mona.

Figure 15 – Lateral and frontal view of the myzaquazo_31ayers.

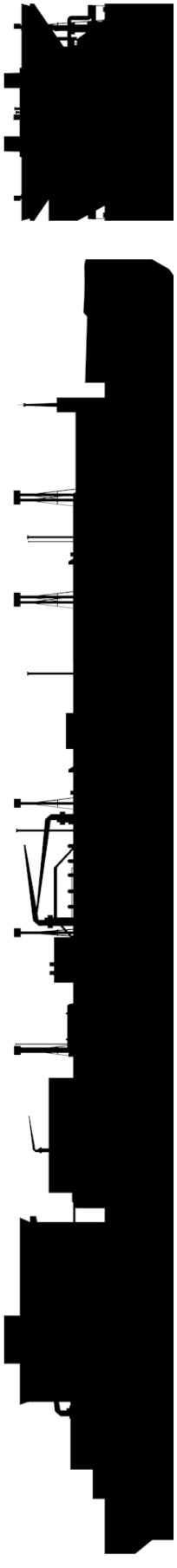


Figure 16 – Lateral and frontal view of the LNG_Yamal_ARC7_299_500.

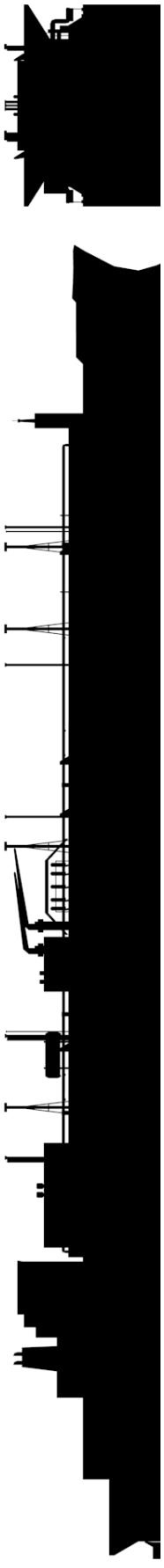


Figure 17 – Lateral and frontal view of the LNG_Conventional_300_460.

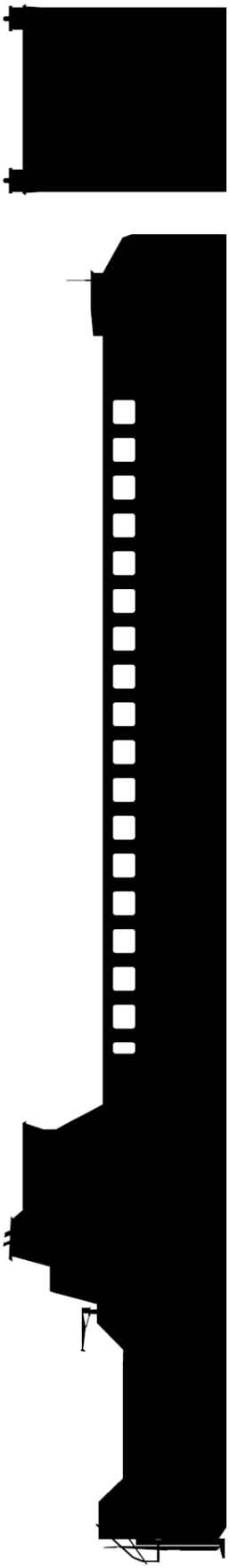


Figure 18 – Lateral and frontal view of the DFDS_JinLing_235_330.

Figure 19 – Lateral and frontal view of the Clementine.

2.6 Conclusions

In this chapter, an automated method is proposed to compute the reference windage areas of the ship mathematical manoeuvring models used in the simulators at FHR. These windage areas are required as input for the computation of wind loads on said ships in order to convert the resultant forces and moments into dimensionless coefficients.

The method uses Blender to render a greyscale image of the front and side view of the ship. The surface areas are computed by counting the number of coloured pixels. The method is fairly efficient, being able to compute the reference areas in a manner of minutes for a range of resolutions. By fitting a four-parameter symmetrical sigmoidal function to the reference areas computed as such, an estimate can be found for an infinite resolution. It is shown that the estimate obtained as such can be significantly more accurate than a value computed at a very fine resolution, but the improvement depends much on the presence of small details. The method shows a second-order trend as a function of the image resolution.

When determining the wind coefficients for a ship model using CFD, the resultant forces are converted to dimensionless quantities by dividing them by dynamic pressure and the reference area. There is no point in doing this if the reference area used in the simulator is not correct. Given the sometimes significant differences between the reference areas as used in the simulator and those computed in this report, it may be necessary to perform a quality check of the reference areas of all vessels as used in the simulator.

Although it is demonstrated that the method gives correct results for sufficiently high resolutions, the convergence is rather slow. This is in part caused by the inexact fit of the image to the vessel dimensions: due to the use of square pixels, it is possible (and likely) that only the first dimension (ship length) exactly matches the requested number of pixels. If the fraction on the right-hand side of Eq. 29 is a floating point number instead of an exact integer, the rounding up of this fraction results in a slightly taller image than the vessel. This then also holds for the image width of the frontal area render. With the use of non-square pixels the rounding would not be necessary, and the results may converge faster to the correct value.

3 Mesh wrapping

3.1 Introduction

As briefly mentioned in Chapter 2, the step of creating a watertight mesh from an initial triangular hull geometry (as used in the simulator for example) may take a significant amount of time when tackled by hand. This is caused by different requirements for the simulator and for CFD applications: the former should contain as few triangles as possible (for good rendering performance) and it is allowed to be non-manifold. Geometry for the latter applications must be watertight to distinguish the inside from the outside. During the evaluation of StarCCM+ in project 18_030, it became clear that StarCCM+ includes a very good surface wrapper as part of its mesh preparation tools (Van Hoydonck *et al.*, 2019). A tool similar to this surface wrapper would result in a potentially significant reduction in time for the preparation of simulator ship geometries for use in CFD wind coefficient computations.

3.2 Available surface wrapping tools

3.2.1 Meshmixer

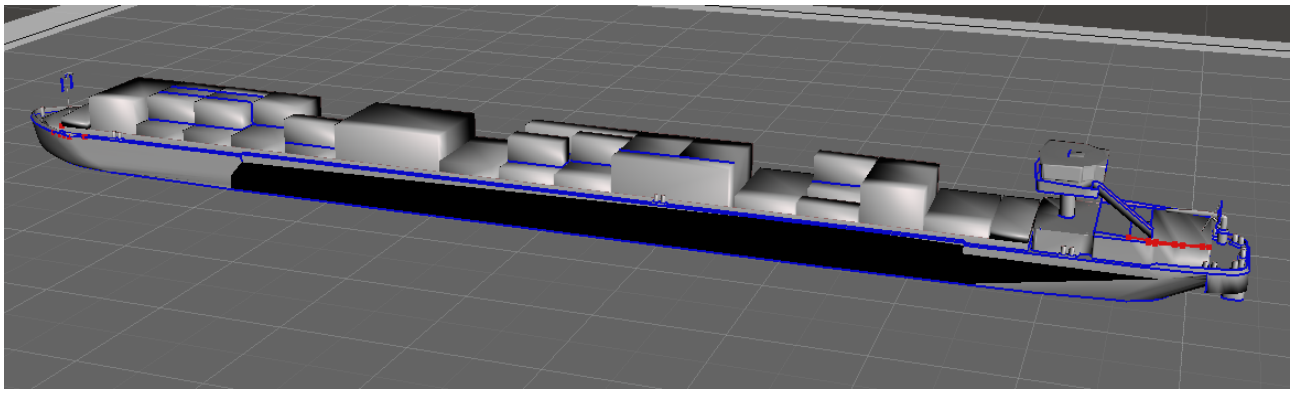
Meshmixer[5] is Autodesk's free software for preparation of geometries for 3D printing. In the *Edit Menu*, there is a tool called *Make Solid*. This tool creates an internal voxel representation of the input shape to create an unambiguous solid model from the input. The result is an approximation, where the biggest drawback is the fixed size of the resulting faces of the mesh approximation. This size is governed by the size of the voxels: the smaller the dimensions of the voxels as compared to the input geometry, the more accurate the final approximation. Details smaller than a few voxels will be lost and sharp edges may be smoothed. Reducing the voxel size has a detrimental effect on the computation time.

In Fig. 20a, the STL input geometry of the Myzaquazo is shown, while Figs. 20b to 20d show the resulting solid model wrapped with an increasingly more detailed mesh. A detail of the steering house is shown in Fig. 21, where it is clear that the floor and railing around the steering house are not properly captured by the tool. Given that this occurs at the finest possible settings, the controls of Meshmixer do not seem to to have enough range to execute the task at hand successfully.

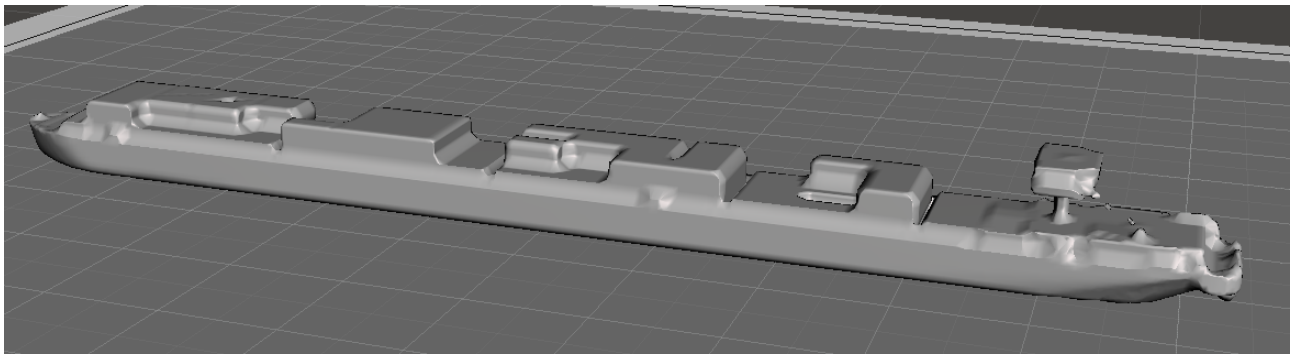
3.2.2 Open Cascade CAD Processor

The commercial branch of Open Cascade markets a CAD processor tool for 3D data preparation and simplification for various applications[6] that includes a shrink wrapping algorithm capable of wrapping any CAD geometry (either Non-Uniform Rational B-Spline (NURBS)-based or polygonal-based) with a watertight unstructured grid that is simulation-ready. A trial version of the software can be downloaded, but this has not been pursued at this moment.

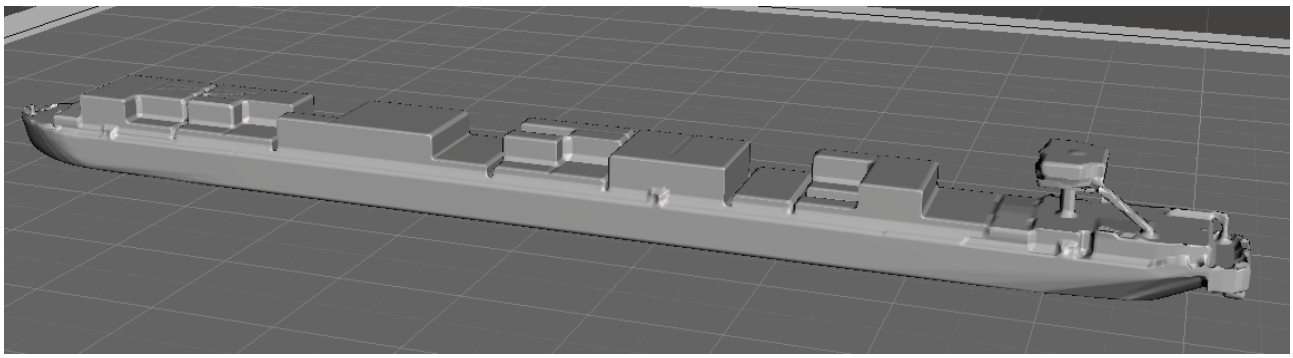
[5] `https://www.meshmixer.com`
[6] `http://www.opencascade.com/products/cad-processor/`



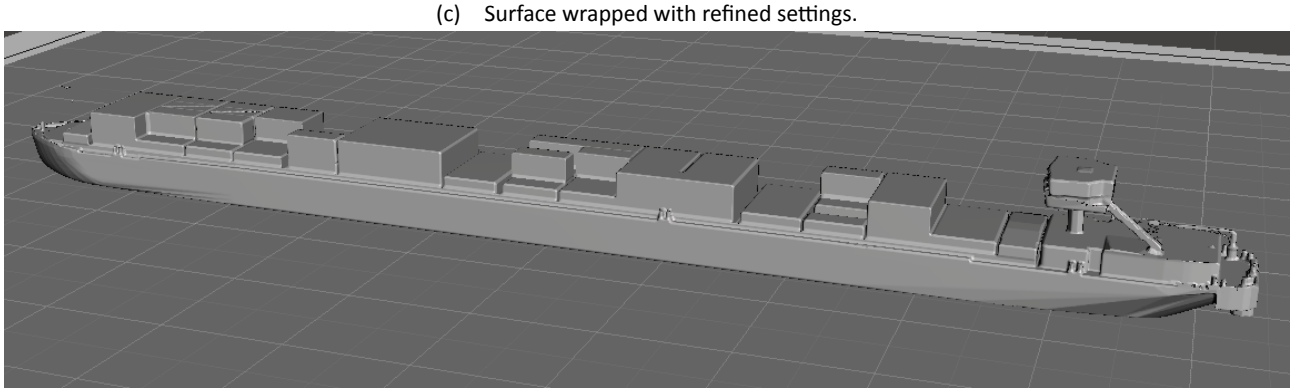
(a) Original STL geometry.



(b) Surface wrapped with the default settings.



(c) Surface wrapped with refined settings.



(d) Surface wrapped with best (slowest) settings.

Figure 20 – Surface wrapping the Myzaquazo STL geometry with Autodesk's Meshmixer.

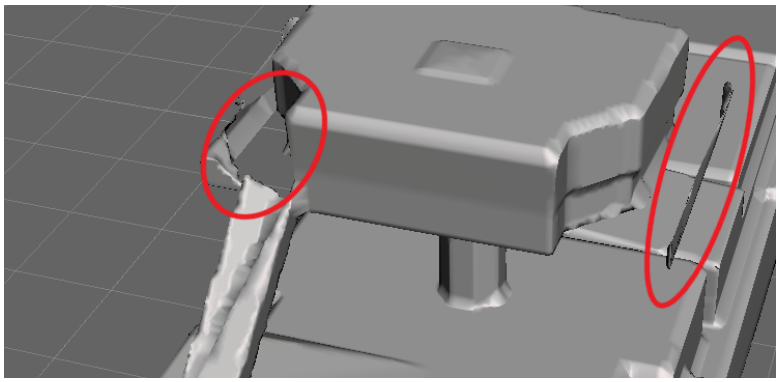


Figure 21 – Issues with the resulting solid model of the steering house of the Myzaquazo at the finest settings. The circled regions show loose geometry (right) and gaps in the model (right).

3.2.3 Blender Remesh Modifier

Starting from Blender 2.81, the Sculpt Remesher tool[7] has been added as an option to the *Remesh* modifier, where it can be accessed as *Voxel* mode, see Fig. 22. As a result, there are now four modes for the *Remesh* modifier:

- Blocks
- Smooth
- Sharp
- Voxel

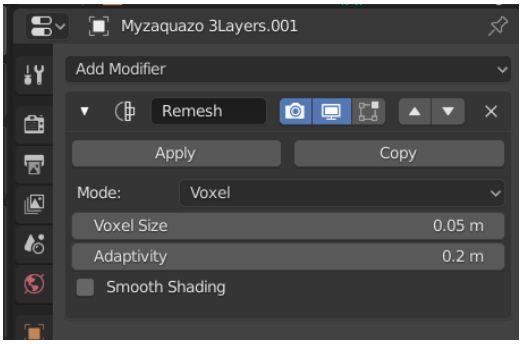


Figure 22 – *Remesh* modifier in *Voxel* mode in Blender.

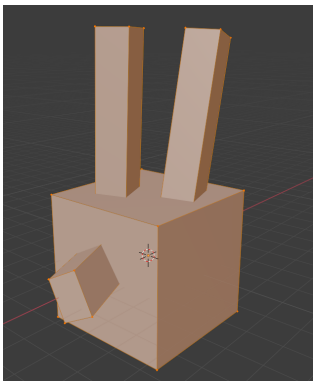
The voxel size is an absolute length with a lower limit of 0.0001 m, which is enough to capture sufficient detail of the original geometry for the purpose of CFD computations. The tool has an *Adaptivity* setting to increase the face sizes in areas where detail is not required such as away from sharp edges that demarcate adjacent non-coplanar surfaces.

For simple geometries consisting of overlapping or intersecting parts that are not aligned to the global axes, the *Remesh* modifier in *Sharp* mode can generate good results without increasing the cell count significantly. This is shown in Fig. 23 where four partially overlapping boxes in the same object are joined while preserving the sharp intersections between the different boxes.

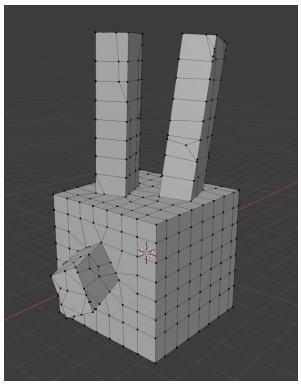
However, for increasingly larger and complex models, extra work is required to ensure that the model is reasonably watertight. Thin features such as railings and gaps between container stacks require a high value for the *Octree depth*, in the case of the Myzaquazo, a depth of at least 11 or even 12 is

[7] Underneath the hood, the remesher tool uses the open-source OpenVDB C++ library (`http://www.openvdb.org`) originally developed at Dreamworks Animation.

required. At this latter depth, the resulting geometry is basically a perfect representation of the initial geometry, but without the imperfections of non-manifold edges. This comes at a price however, because the vertex count is more than 5.772×10^6.



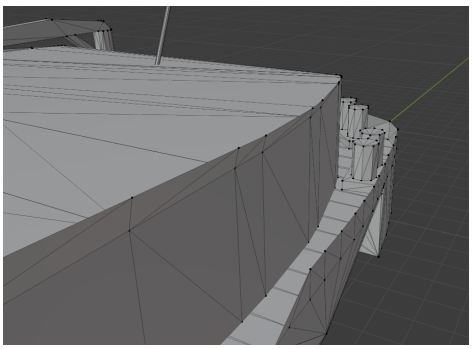
(a) Input geometry.



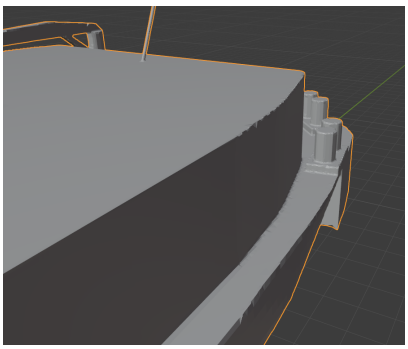
(b) Manifold result.

Figure 23 – Blender's *Remesh* modifier in *Sharp* mode applied to four intersecting boxes.

The requirements for the Remesh modifier in *Voxel* mode are not as strict: holes such as the hull bottom must be filled, but overlapping geometry is allowed, as long as the algorithm can construct a manifold mesh around the input. It does not discretize baffles (thin surfaces) as can be done in *Sharp* mode (see Fig. 24), so the railing at the stern should either be removed, or it should be extruded to give it some thickness (Fig. 25).



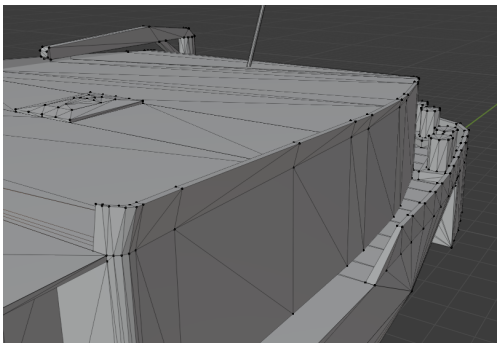
(a) Input geometry.



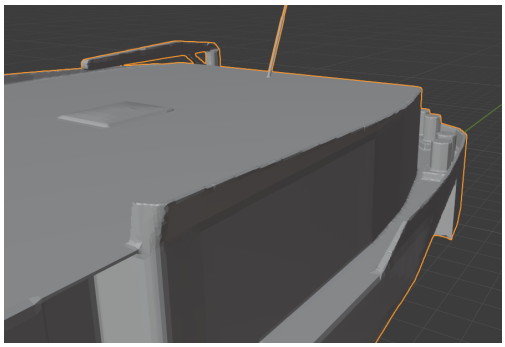
(b) Manifold result with baffles removed.

Figure 24 – Blender's *Remesh* modifier in *Voxel* mode applied to the Myzaquazo hull geometry.

For the geometry of the Myzaquazo, a first attempt at preparing the geometry to create a surface wrapped mesh took less than an hour, which is a significant reduction as compared to the required time that was needed when cleaning the Tripoli hull geometry by hand (multiple days). Fig. 26 shows the geometry of the Myzaquazo near the bow. The flags and some other loose parts were removed, but otherwise all geometry was kept. The voxel size was set to 0.02 m while the *adaptivity* was set to four times that value (0.08 m). With these settings, the resulting wrapped geometry is obtained in approximately one minute. Fig. 26a shows the input geometry, while the surface wrapped results is shown in edit mode in Fig. 26b. With these settings, the mast is captured accurately. The number of vertices is increased significantly (from 2672 to 1 072 518), but the resulting mesh is a perfect manifold and can be used directly in CFD applications.

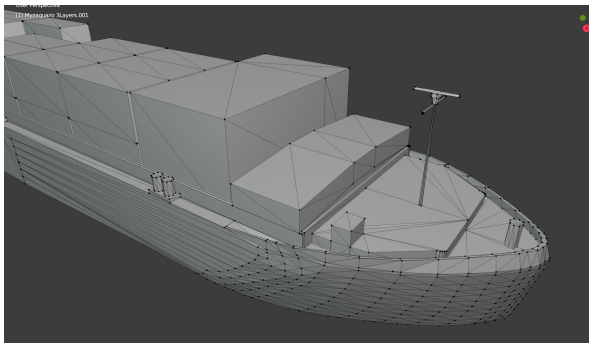


(a) Modified input geometry: baffles with thickness.

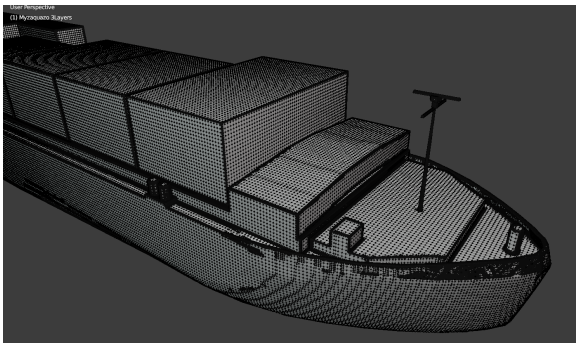


(b) Manifold result with baffles included.

Figure 25 – Blender's *Remesh* modifier in *Voxel* mode applied to a slightly modified Myzaquazo hull geometry.



(a) Input geometry.



(b) Surface wrapped geometry.

Figure 26 – Surface wrapping the Myzaquazo STL geometry with Blender's remesh modifier.

3.3 Conclusions

Using Blender's Remesh modifier in Voxel model in the workflow to compute wind coefficients of a ship will reduce the required preparatory time significantly, which brings regular use of CFD for this type of application again closer. Although the complete geometry with all details can be captured, edge sharpness is not exactly maintained. This could be an issue when the resulting geometry is used in meshing software such as Hexpress or OpenFOAM's snappyHexMesh. However, the intention is to use this type of model in Gerris, where the hull geometry is approximated in an octree. Sharp edges are not guaranteed to be preserved there either, this depends entirely on the number of octree subdivisions used in the computation.

4 Summary and conclusions

In this first report regarding the evaluation of the open-source Gerris flow solver, optimisations are reported related to two preparatory steps that must be executed before actual CFD computations can be started. The first step is related to the computation of the ship frontal and lateral reference areas, while the second step concerns the conversion of the ship geometry as used in the simulators at FHR for use in CFD computations. For both steps, an STL representation of the geometry is used as input, although other polygonal formats are possible as well. The steps make use of the existing modelling and rendering capabilities available in the open-source 3D content creation suite Blender.

For the frontal and lateral surface area computations, the ship frontal and lateral views are rendered as greyscale images. The reference surface areas follow from the number of coloured pixels in the resulting renders, the total pixel count in the renders and the dimensions of the models. The method can compute the reference areas in a manner of minutes for a range of image resolutions. An extrapolation to an infinite resolution is implemented by fitting a four-parameter symmetrical sigmoidal function to the computed reference areas. It is shown that the estimate obtained as such can be significantly more accurate (and quicker) than a value computed at an extremely fine resolution, but the improvement depends much on the presence of small details.

For a small number of vessels as used in the simulator, the frontal and lateral surface areas at specific draft values are computed and compared with the values as recorded in the `load_*xml` files of the ships. For some of these, the differences are rather large between the reference areas as used in the simulator and those computed in this report. This can be caused by imperfections in the 3D models: a small error in deck height in the 3D model can result in a significant difference in the windage area. The reference areas of the models as used in the `load_*xml` files are also not always computed from the 3D models: some are based on general arrangement plans or information from the ship owner.

Preparing a 3D geometry as used in the simulator for CFD computations is a task that requires a significant amount of time when executed by hand. The evaluation of StarCCM+ (project 18_030) showed that a significant improvement in efficiency can be obtained with the help of an automatic surface wrapping tool. Recent version of Blender link against the OpenVDB library, which makes available an efficient method to wrap an existing CAD model with a watertight shell in Blender. It is estimated that – on average – the optimisations for the steps as currently reported reduce the preparation time from up to two days to as little as two hours.

References

Andersen, **I. M. V.** (2007). Vindkræfter på Containerskibe. (M.Sc. thesis). Institut for Mekanik, Energi og Konstruktion, Danmarks Tekniske Universitet

Andersen, **I. M. V.** (2013). Wind loads on post-panamax container ship. *Ocean Engineering 58*: 115–134. DOI: https://doi.org/10.1016/j.oceaneng.2012.10.008

Blendermann, **W.** (2013). Practical ship and offshore structure aerodynamics. Hamburg University of Technology: Hamburg, Germany. ISBN: 978-3-89220-669-9

Blendermann, **W.** (1993). Schiffsform und Windlast-Korrelations- und Regressionsanalyse von Windkanalmessungen am Modell. 533. Hamburg, Germany

Blocken, **B.; Stathopoulos**, **T.; Carmeliet**, **J.** (2007). CFD simulation of the atmospheric boundary layer: wall function problems. *Atmospheric Environment 41 (2)*: 238–252. DOI: https://doi.org/10.1016/j.atmosenv.2006.08.019

Fujiwara, **T.; Nimura**, **T.** (2005). New estimation method of wind forces acting on ships on the basis of mathematical model. ***in**: Proceedings of the Fifteenth (2005) International Offshore and Polar Engineering Conference*. Seoul, Korea, 19th–24th June 2005. The International Society of Offshore and Polar Engineers. ISBN: 1880653648. pp. 82–89

Janssen, **W. D.; Blocken**, **B.; van Wijhe**, **H. J.** (2017). CFD simulations of wind loads on a container ship: Validation and impact of geometrical simplifications. *Journal of Wind Engineering and Industrial Aerodynamics 166*: 106–116. DOI: https://doi.org/10.1016/j.jweia.2017.03.015

Richards, **P. J.; Norris**, **S. E.** (2012). Appropriate boundary conditions for a pressure driven boundary layer. ***in**:* Brandner, P. A.; Pearce, B. W. (Eds.). *Proceedings of the 18th Australasian Fluid Mechanics Conference*. Launceston, Australia, 3rd–7th December 2012. Australasian Fluid Mechanics Society. ISBN: 978-0-646-58373-0. p. 4

Ueno, **M.; Kitamura**, **F.; Sogihnara**, **N.; Fujiwara**, **T.** (2012). A simple method to estimate wind loads on ships. ***in**: The 2012 World Congress on Advances in Civil, Environmental, and Materials Research (ACEM' 12)*. Seoul, Korea, 26th August 2012–30th August 2005. International Association of Structural Engineering and Mechanics. pp. 2314–2322

Van Hoydonck, **W.; Delefortrie**, **G.; Peeters**, **P.; Mostaert**, **F.** (2015a). Computation of aerodynamic coefficients of ships using FINE/Marine: initial evaluation. 3.0. *WL Rapporten*, 12_106. Antwerp, Belgium

Van Hoydonck, **W.; Delefortrie**, **G.; Vos**, **S.; Peeters**, **P.; Mostaert**, **F.** (2015b). Computation of aerodynamic coefficients of ships using FINE/Marine: Wind coefficients for the Tripoli estuary container vessel at a draft of 3.5 m. 3.0. *WL Rapporten*, 12_106. Antwerp, Belgium

Van Hoydonck, **W.; Eloot**, **K.; Verelst**, **K.; Mostaert**, **F.** (2019). Evaluation of the CFD Package StarCCM+. 2.0. *WL Rapporten*, 18_030. Antwerp, Belgium

Van Hoydonck, **W.; Vos**, **S.; Peeters**, **P.; Mostaert**, **F.** (2016). CFD computations of the wind climate behind a roro ship. Version 3.0. *FHR Reports*, 15_096. Flanders Hydraulics Research: Antwerp, Belgium

Van Zwijnsvoorde, **T.; Donatini**, **L.; Hoydonck**, **W. V.; Lataire**, **E.** (2019). Wind modeling for large container vessels: a critical review of the calculation procedure. *International Journal of Transport Development and Integration 3 (4)*: 369–381. DOI: 10.2495/TDI-V3-N4-369-381

Wagner, **M.** (2005). Wind-Tunnel Tests with FSG NB 731/732. FORCE 2005031. FORCE Technology: Lyngby, Denmark

DEPARTMENT **MOBILITY & PUBLIC WORKS**
Flanders hydraulics Research

Berchemlei 115, 2140 Antwerp
T +32 (0)3 224 60 35
F +32 (0)3 224 60 36
waterbouwkundiglabo@vlaanderen.be
www.flandershydraulicsresearch.be